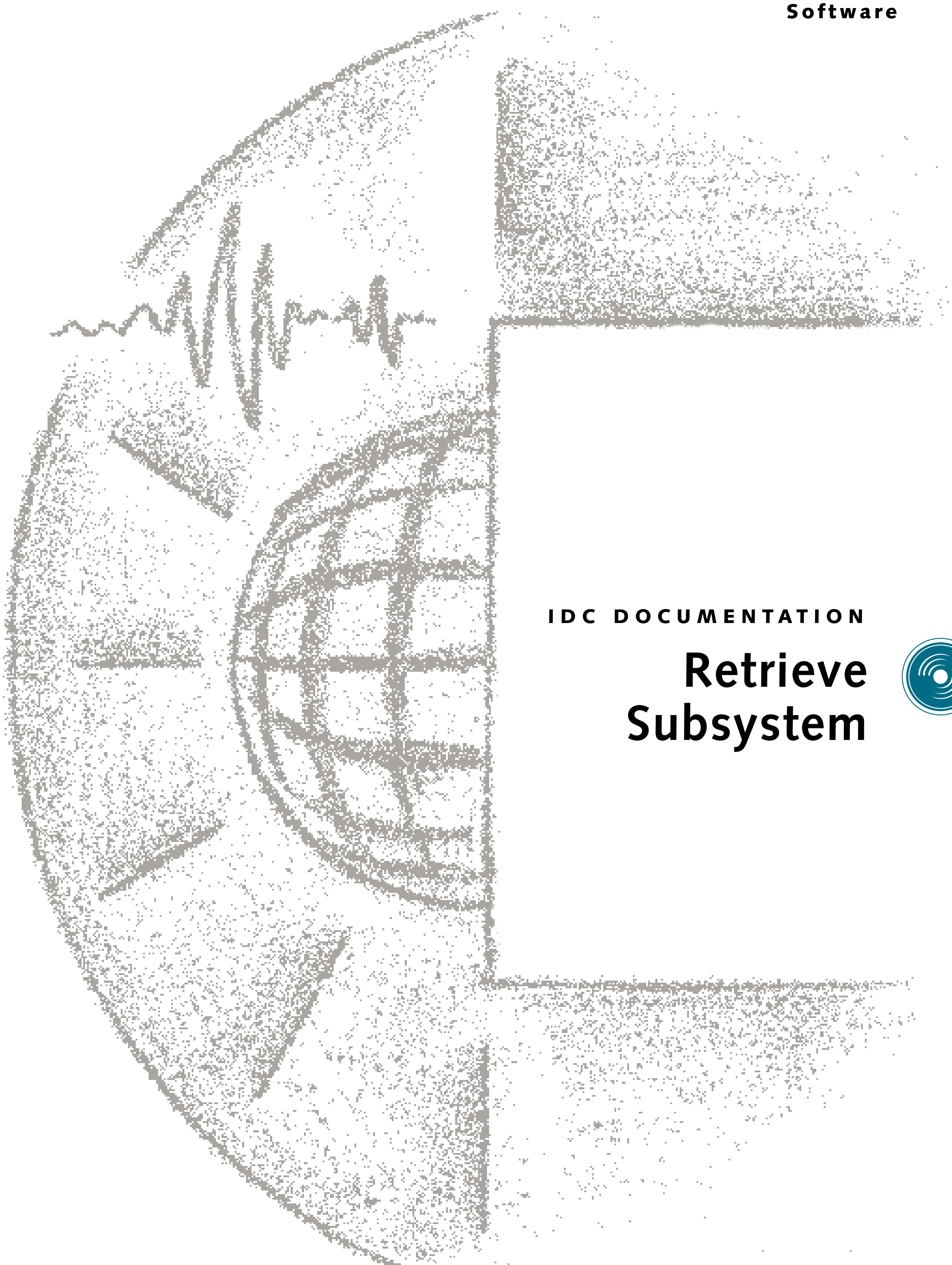


Software

IDC DOCUMENTATION

Retrieve Subsystem

Notice

This document was published August 2001 by the Monitoring Systems Operation of Science Applications International Corporation (SAIC) as part of the International Data Centre (IDC) Documentation. Every effort was made to ensure that the information in this document was accurate at the time of publication. However, information is subject to change.

Contributors

Alexander J. Boresch, Science Applications International Corporation

Ordering Information

The ordering number for this document is SAIC-01/3024.

This document is cited within other IDC documents as [IDC7.4.3].

Retrieve Subsystem

CONTENTS

Retrieve Subsystem

FIGURES

Retrieve Subsystem

TABLES

About this Document

This chapter describes the organization and content of the document and includes the following topics:

- Purpose
- Scope
- Audience
- Related Information
- Using this Document

About this Document

PURPOSE

This document describes the design and requirements of the Retrieve Subsystem software of the International Data Centre (IDC). The software is a computer software component (CSC) of the Data Services Computer Software Configuration Item (CSCI). This document provides a basis for implementing, supporting, and testing the software.

SCOPE

This document describes the architectural and detailed design of the software including its functionality, components, data structures, high-level interfaces, method of execution, and underlying hardware. This information is modeled on the Data Item Description for *Software Design Descriptions* [DOD94a].

AUDIENCE

This document is intended for all engineering and management staff concerned with the design and requirements of all IDC software in general and of the Retrieve Subsystem in particular. The detailed descriptions are intended for programmers who will be developing, testing, or maintaining the Retrieve Subsystem.

RELATED INFORMATION

The following documents complement this document:

- *Database Schema* [IDC5.1.1Rev2]
- *Formats and Protocols for Messages* [IDC3.4.1Rev2]

See "References" on page 33 for a list of documents that supplement this document. The following UNIX manual (man) pages apply to the existing Retrieve Subsystem software:

- *MessageSend*
- *WaveAlert*
- *dispatch*

USING THIS DOCUMENT

This document is part of the overall documentation architecture for the IDC. It is part of the Software category, which describes the design of the software. This document is organized as follows:

- Chapter 1: Overview

  This chapter provides a high-level view of the Retrieve Subsystem, including its functionality, components, background, status of development, and current operating environment.
- Chapter 2: Architectural Design

  This chapter describes the architectural design of the Retrieve Subsystem, including its conceptual design, design decisions, functions, and interface design.
- Chapter 3: Detailed Design

  This chapter describes the detailed design of the Retrieve Subsystem including its data flow, software units, and database design.
- References

  This section lists the sources cited in this document.
- Glossary

  This section defines the terms, abbreviations, and acronyms used in this document.

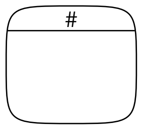
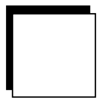
- Index

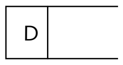
  This section lists topics and features provided in the document along with page numbers for reference.

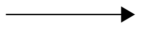
Conventions

This document uses a variety of conventions, which are described in the following tables. Table I shows the conventions for data flow diagrams. Table II shows the conventions for entity-relationship (E-R) diagrams. Table III lists typographical conventions.

TABLE I: DATA FLOW SYMBOLS

Description	Symbol[1]
process	
external source or sink of data	

data store
D = disk store
Db = database store
	

data flow	

1. Symbols in this table are based on Gane-Sarson conventions [Gan79].

TABLE II: ENTITY-RELATIONSHIP SYMBOLS

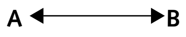
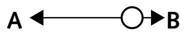
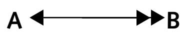
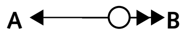
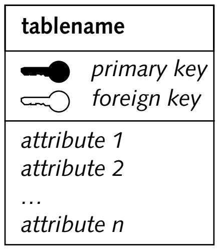
Description	Symbol
One **A** maps to one **B**.	
One **A** maps to zero or one **B.**	
One **A** maps to many **B**s.	
One **A** maps to zero or many **B**s.	
database table	

TABLE III: TYPOGRAPHICAL CONVENTIONS

Element	Font	Example
database table	**bold**	**request**
database attributes	*italics*	*status*
processes, software units, and libraries		*WaveExpert*
user-defined arguments and variables used in parameter (par) files or program command lines		`delete-remarks` *object*
titles of documents		*Database Schema*
computer code and output	`courier`	`PENDING`
filenames, directories, and Web sites		`MSGDIR`

Chapter 1: Overview

This chapter provides a general overview of the Retrieve Subsystem and includes the following topics:

- Introduction
- Functionality
- Identification
- Status of Development
- Background and History
- Operating Environment

Chapter 1: Overview

INTRODUCTION

The software of the IDC acquires time-series and radionuclide data from stations of the International Monitoring System (IMS) and other locations. These data are passed through a number of automatic and interactive analysis stages, which culminate in the estimation of location and origin time of events (earthquakes, volcanic eruptions, and so on) in the earth, including its oceans and atmosphere. The results of the analysis are distributed to States Parties and other users by various means. Approximately one million lines of developmental software are spread across six CSCIs of the software architecture. One additional CSCI is devoted to run-time data of the software. Figure 1 shows the logical organization of the IDC software. The Data Services CSCI receives, archives, and distributes data through the following CSCs:

- Continuous Data Subsystem

  This software acquires time-series data according to standard protocols and forwards the data to external users ([IDC3.4.2] and [IDC3.4.3]).

- Message Subsystem

  This software exchanges data in response to user requests. The data are formatted according to a standard and exchanged through UNIX mail (see [IDC3.4.1Rev2]). This software also provides the interface to mail for the Retrieve and Subscription Subsystems.

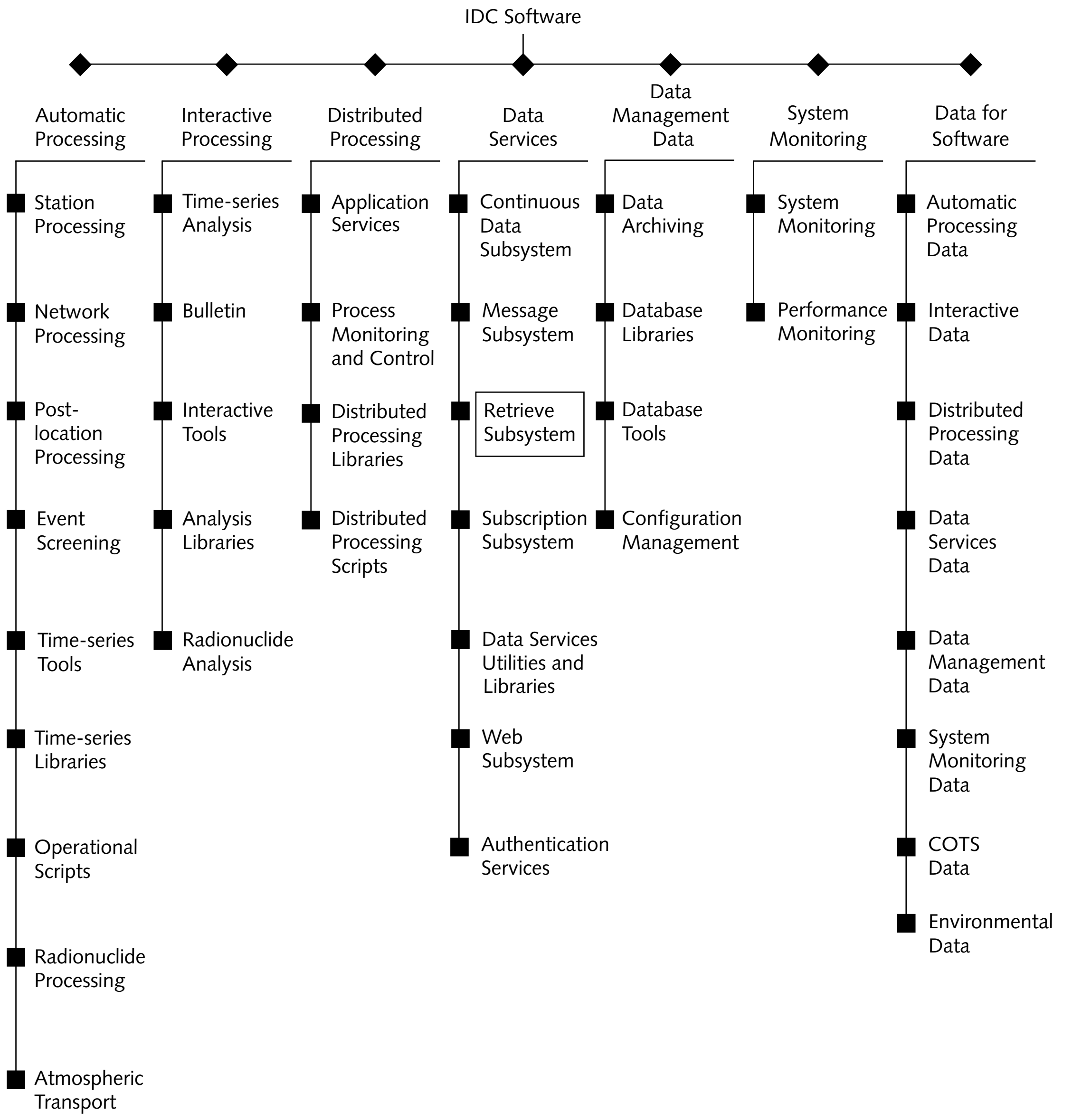


FIGURE 1. IDC SOFTWARE CONFIGURATION HIERARCHY

- Retrieve Subsystem

  This software prepares messages, formatted according to the standard, that retrieve segments of data from stations of the IMS auxiliary seismic network (see [IDC3.4.1Rev2]). The Message Subsystem software exchanges the messages and parses the response messages. The Retrieve Subsystem reconciles the received data with the requests.

- Subscription Subsystem

  This software maintains a subscriber database and prepares the regular data products for delivery to subscribers. The Message Subsystem receives the subscription requests and delivers the subscription products.

- Data Services Utilities and Libraries

  This software consists of utilities used by data services operators and libraries common to data services.

- Web Subsystem

  This software runs the IDC website.

- Authentication Services

  This software provides data signing and verification services to Data Services subsystems using the Digital Signature Algorithm (DSA).

Figure 2 shows the relationship of the Retrieve Subsystem to the other components of the Data Services CSCI. The Retrieve Subsystem fulfills the following roles: It collects requests for auxiliary seismic data from the Automatic Processing CSCI, the analyst tool *Interactive Auxiliary Data Request* (*IADR*) via *WaveExpert*, and from the Message Subsystem (indirect user requests via *AutoDRM*). It uses the Message Subsystem [IDC7.4.2] to send request messages to auxiliary seismic stations. It also creates short sample requests to check auxiliary station availability. Finally, it verifies the success of data retrieval and resends requests if no data have been received. The dataflow is controlled by the Distributed Application Control System (DACS), and log files are written to keep track of the activities of the Retrieve Subsystem.

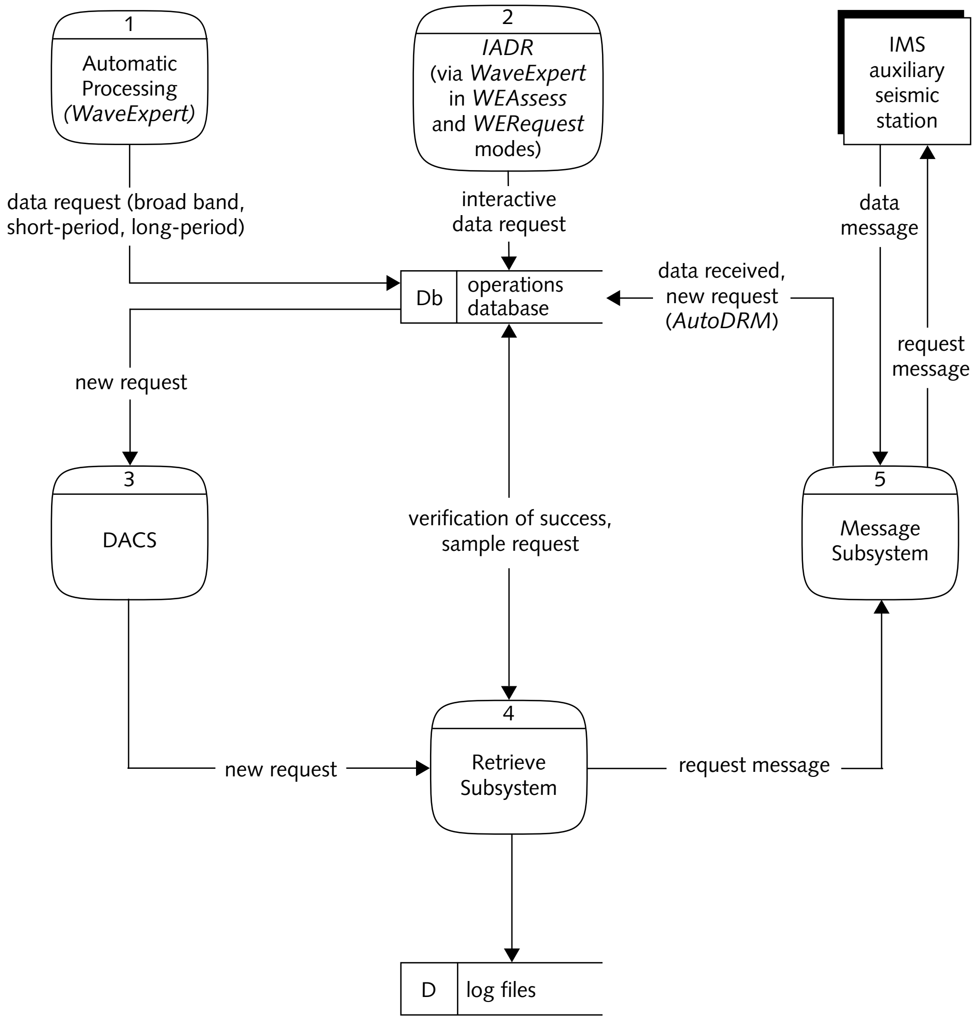

1
Automatic Processing (WaveExpert)
2
IADR (via WaveExpert in WEAssess and WERequest modes)
IMS auxiliary seismic station
data request (broad band, short-period, long-period)
interactive data request
data message
data received, new request (AutoDRM)
Db
operations database
request message
new request
3
DACS
verification of success, sample request
5
Message Subsystem
4
Retrieve Subsystem
new request
request message
D
log files

FIGURE 2. RELATIONSHIP OF RETRIEVE SUBSYSTEM TO OTHER SOFTWARE UNITS OF DATA SERVICES CSCI

FUNCTIONALITY

The Retrieve Subsystem requests data from auxiliary seismic stations. Request messages are passed to the Message Subsystem [IDC7.4.2], which sends them to IMS stations, receives data messages back from the stations, and parses data into the database and auxiliary waveform data store. The Retrieve Subsystem verifies that the requested data have been successfully received and parsed. If no data have been received, the Retrieve Subsystem sends the request message again. Five types of auxiliary data requests are made. Automatic Processing requests additional data via *WaveExpert* to improve event location and magnitude estimation for events in SEL1 (Standard Event List) and SEL2. The DACS is configured to request all available long-period data from auxiliary stations. The seismic analysts can request auxiliary data via the tool *IADR*, which uses *WaveExpert* in the *WEAssess* and *WERequest* modes. *AutoDRM* inserts redirected data requests from external users if the data are not available at the data center. Finally, the Request Subsystem itself creates short sample requests for all auxiliary stations to check station availability.

IDENTIFICATION

The Retrieve Subsystem's components are identified as follows:

- *polling*
- *dispatch*
- *MessageSend*
- *WaveAlert*

STATUS OF DEVELOPMENT

The Retrieve Subsystem is completely developed. A recently added element of the IDC system is the long-period data retrieve pipeline, which has been added in the Distributed Processing CSCI to retrieve all available long-period data from auxiliary seismic stations. Although these requests are passed via the Retrieve Subsystem to the Message Subsystem, this addition was a configuration change in a different

CSCI, and the Retrieve Subsystem has remained unchanged. Another recent addition is the ability to redirect data requests from external users to the IMS stations if the data are not available at the data center. The Message Subsystem (*AutoDRM*) creates entries for such requests in the **request** table and the Retrieve Subsystem processes them as any other request. When the data are retrieved, the Retrieve Subsystem sends a message to *AutoDRM* so that *AutoDRM* can respond to the external user. *WaveAlert* has been enhanced to send this message to *AutoDRM*.

BACKGROUND AND HISTORY

David Corley, John Coyne, and Dana Dickinson of the Center for Monitoring Research (CMR) developed the Retrieve Subsystem in 1993. It was later revised by Yu-Long Kung and David Salzberg, both of the CMR. *dispatch* was developed by Benjamin Kohl at the CMR in Arlington, Virginia, U.S.A, in 1997, and *polling* was developed by Richard Alger in 1994 and revised in the following years by several other staff members of the CMR.

The Retrieve Subsystem was first used operationally in January 1995 under the GSETT-3 program. It currently is an element of the Prototype International Data Center (PIDC) at the CMR.

The International Data Centre of the Comprehensive Nuclear-Test-Ban Treaty Organization (CTBTO IDC) in Vienna, Austria also has installed the Retrieve Subsystem software.

OPERATING ENVIRONMENT

The following paragraphs describe the hardware and commercial-off-the-shelf (COTS) software required to operate the Retrieve Subsystem.

Hardware

The Retrieve Subsystem is designed to run on a UNIX workstation such as the SPARCstation 20/612. Typically, the hardware is configured with 64 MB of memory and a minimum of 2 GB of magnetic disk. The Retrieve Subsystem must obtain other services (such as database access and Interprocess Communication [IPC] resources) over its Ethernet connection to other computers. Figure 3 shows a representative hardware configuration.

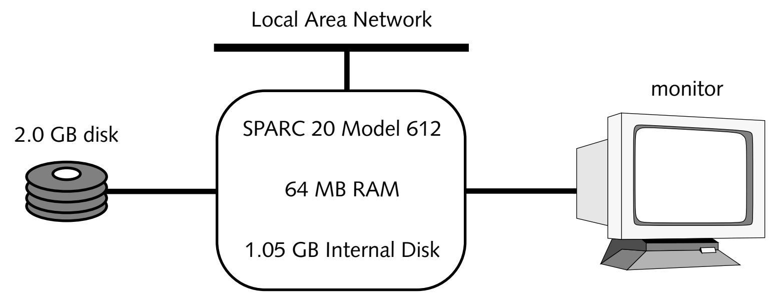


FIGURE 3. REPRESENTATIVE HARDWARE CONFIGURATION FOR RETRIEVE SUBSYSTEM

Commercial-Off-The-Shelf Software

The software is designed for Solaris 2.7 and ORACLE 8i. The Retrieve Subsystem is controlled by the DACS, which needs Tuxedo 6.5. IPC resources need to be configured for Tuxedo. See [IDC6.5.2Rev0.1] for instructions on DACS configuration.

Chapter 2: Architectural Design

This chapter describes the architectural design of the Retrieve Subsystem and includes the following topics:

- Conceptual Design
- Design Decisions
- Functional Description
- Interface Design

Chapter 2: Architectural Design

CONCEPTUAL DESIGN

The Retrieve Subsystem provides the functionality to automatically retrieve data from IMS auxiliary seismic stations. These data are used by the Automatic Processing CSCI to improve results of Global Association, Event Location Processing, and Post-location Processing (magnitude calculation) for detected events. The Retrieve Subsystem is designed to read requests that may be written by different requestors from the database, to format request messages in the GSE2.0 (Group of Scientific Experts) and IMS1.0 formats [IDC3.4.1Rev2], and pass these messages to the Message Subsystem. The Retrieve Subsystem also verifies the success of data retrieval by comparing received data with the original data requests. If no data have been received, the requests are automatically retried. Another module of the Retrieve Subsystem performs regular checks of auxiliary station availability by creating short sample requests. Other existing subsystems like the Automatic Processing CSCI, the Interactive CSCI, the Distributed Processing CSCI, and especially the Message Subsystem [IDC7.4.2] are integrated as far as possible to perform these tasks. Database tables and message files serve as main interfaces to other subsystems. No direct interface with the operator is necessary, because the Retrieve Subsystem is designed to run automatically. However, the operator can monitor the performance of the Retrieve Subsystem by checking the log files of the individual applications and database tables. An additional tool for monitoring the **request** table is *RequestFlow,* which is an element of the Distributed Processing CSCI.

DESIGN DECISIONS

The following design decisions pertain to the Retrieve Subsystem.

Programming Language

MessageSend and *WaveAlert* are written in the C programming language. *dispatch* and *polling* are written in the Perl scripting language.

Global Libraries

The software of the Retrieve Subsystem is linked to the following developmental libraries: *libconvert*, *libgdi*, *libgeog*, *ibgsemsg*, *libgsewf*, *liblogout*, *libmisc*, *libpar*, *libstdtime*, *libtable*, and *libwfm.*

The software of the Retrieve Subsystem is also linked to the following COTS libraries: *libdl*, *libF77*, *libm*, *libnsl*, and *libsocket.*

Database

The Retrieve Subsystem obtains data from and writes data to the ORACLE database. The Retrieve Subsystem uses the database tables **request, msgdisc, msgdest, outage, wfdisc,** and **sitepoll**. The **request** table serves as the main input source for the Retrieve Subsystem; *polling* uses the **sitepoll** table to obtain information about current auxiliary stations and channels; the **msgdisc** and **msgdest** tables serve as interfaces to the Message Subsystem for outgoing request messages; and the **wfdisc** and **outage** tables contain records about received data and station outages and are compared to the **request** table by *WaveAlert*.

Interprocess Communication (IPC)

The Retrieve Subsystem does not use IPC directly. However, *dispatch* is controlled by the DACS through *WaveGet_server* and *tuxshell,* which call *dispatch* and provide the input from the **request** table. The DACS heavily uses IPC resources, which must be configured accordingly. See the *DACS Software User Manual* [IDC6.5.2Rev0.1] for further information.

Filesystem

The filesystem holds the run-time parameters of the Retrieve Subsystem (par files) with the exception of *polling,* which does not use a parameter file. *MessageSend* also writes formatted messages to the filesystem. Descriptors of these files are stored in the ORACLE database. Each program's log file is written to the filesystem, again with the exception of *polling,* which does not write a log file.

UNIX Mail

Although messages are received and delivered via UNIX mail by the Message Subsystem, the Retrieve Subsystem itself does not use UNIX mail. However, *polling* is controlled from *cron*, which uses UNIX mail to send a notification to the UNIX user if the *cron* job cannot be executed.

Web

While the Message Subsystem uses the Web for some incoming request messages from external users of IDC data and products, the Retrieve Subsystem has been exclusively designed to handle outgoing request messages and monitor the rate of successfully received data for those requests. Therefore, the Retrieve Subsystem does not use the Web.

Design Model

The design of the Retrieve Subsystem is primarily influenced by timeliness, flexibility, and reliability requirements. Although the system must send request messages in a timely manner, a limited delay is acceptable to allow for IPC. Timeliness and the success rate of data retrieval are mainly dependent on availability of IMS stations and the communication links to the stations. Reliability and long-term tracking of data requests and messages are also critical requirements. Therefore, the system uses the IDC's database for both IPC and to archive processing information about requested and received data.

Database Schema Overview

The Retrieve Subsystem uses the ORACLE database for the following purposes:

- reading requests that have been issued from different requestors
- updating the *state* of requests
- writing descriptive information about outgoing messages
- verifying the success of data retrieval
- reading information about stations that should be checked for availability

Table 1 shows the tables used by the Retrieve Subsystem. The Name field identifies the database table. The Mode field is "R" if the Retrieve Subsystem reads from the table and "W" if the system writes to the table.

TABLE 1: DATABASE TABLES USED BY RETRIEVE SUBSYSTEM

Name	Mode	Description
request	R/W	holds information needed to retrieve data from stations of the auxiliary network
sitepoll	R	holds information about stations to be checked for availability
msgdest	W	holds information about messages sent from the IDC
msgdisc	W	holds message information including date and time the message was sent and received, and a message file descriptor
outage	R	holds information about time intervals for which stations have not been available
wfdisc	R	holds information about received data including start and end time of the data interval and a waveform file descriptor
lastid	R/W	holds information about the last sequential value of numeric keys used by other tables

FUNCTIONAL DESCRIPTION

The Retrieve Subsystem has three functions: it creates request messages for auxiliary seismic data, it checks auxiliary station availability, and it verifies the success of data retrieval and retries data requests, if necessary.

Figure 4 shows the major functions of the Retrieve Subsystem. Requests are written to the database by external requestors as well as the Retrieve Subsystem, which creates short sample requests to check station availability. The Retrieve Subsystem then reads the requests from the database and formats request messages in the GSE2.0 or IMS1.0 format, which are passed to the Message Subsystem [IDC7.4.2]. The Message Subsystem exchanges request and data messages with the IMS auxiliary stations and records information about received data in the database. Finally, the Retrieve Subsystem compares information about received data with the data requests to verify the success of the data retrieval process. If no data have been received, the request is retried.

Checking Auxiliary Station Availability

Availability of auxiliary stations is checked by sending sample requests for short intervals of one minute length. This process is referred to as polling the auxiliary network. The list of stations and channels to be checked is read from the **sitepoll** table and requests are inserted into the **request** table. Further processing of these sample requests are the same as for requests, which have been inserted by other requestors. The success of data retrieval for these sample requests is taken as a measure for auxiliary station availability.

Creating Request Messages

Requests are read from the **request** table where all data requestors (for example, *WaveExpert*, *polling*, and so forth) insert their auxiliary data requests. The time, station, and channel information of the request entries are processed to format GSE2.0 or IMS1.0 data request messages. The request messages are written to a permanent storage directory, and descriptive information about the messages is inserted into the **msgdisc** and **msgdest** tables, which serve as an interface to the Message Subsystem. The Message Subsystem then picks up the data requests and sends them via UNIX mail to the IMS stations. Responses from the IMS stations are again parsed by the Message Subsystem, received data are stored in the filesystem, and information about received data intervals is inserted into the **wfdisc** table.

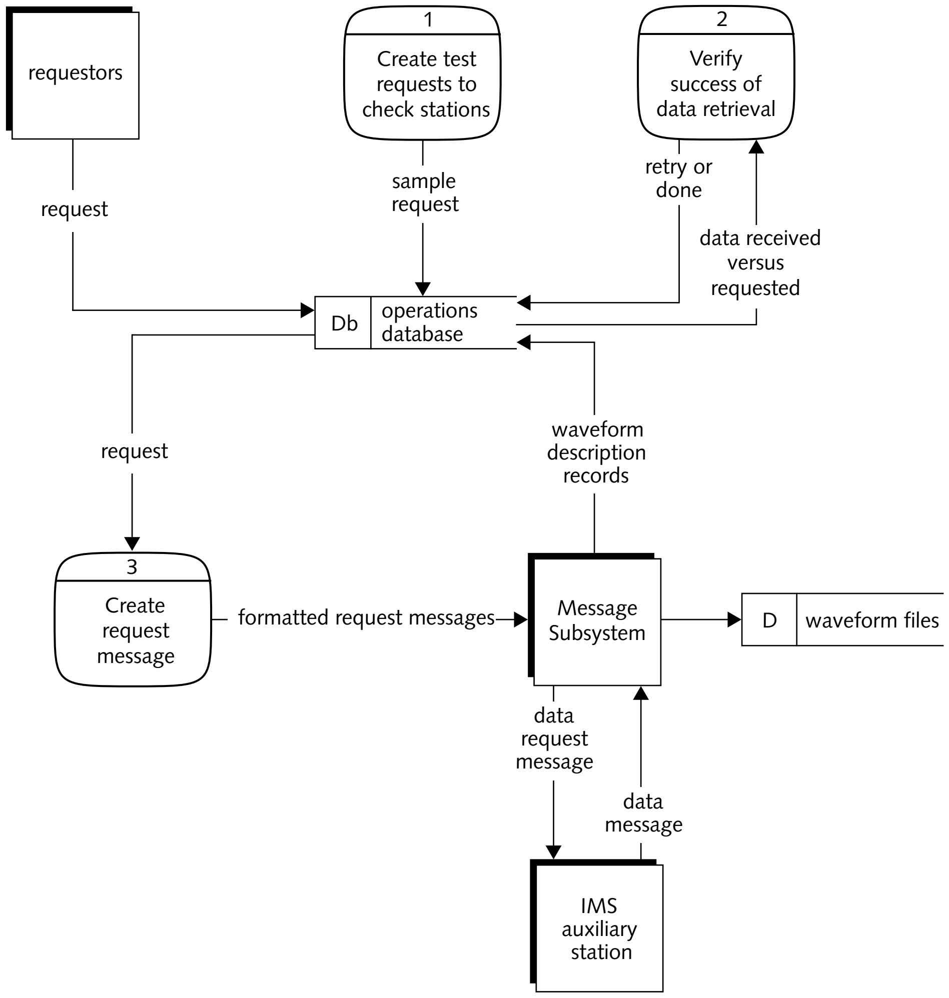


FIGURE 4. FUNCTIONAL DESIGN OF RETRIEVE SUBSYSTEM

Verifying the Success of Data Retrieval

Verification of data retrieval is done by comparing information in the **wfdisc** and **outage** tables to the **request** table. If no data are received after a configurable time interval (*cleanup-time*), the request is re-submitted to the IMS station. Retry attempts to obtain the data continue, but if no data are received after another configurable time interval (*max-submit-time*), the Retrieve Subsystem stops processing the request. The Automatic Processing subsystem has access to the data as soon as they have been received and parsed by the Message Subsystem. If the request was from *AutoDRM* (indirect user request) a message is sent to *AutoDRM* so it can respond to the external user. Information about received data is stored in the **wfdisc** table.

INTERFACE DESIGN

This section describes the Retrieve Subsystem's interfaces with other IDC systems, external users, and operators.

Interface with Other IDC Systems

The Retrieve Subsystem receives requests from data requestors (the Automatic and Interactive Processing CSCIs, *AutoDRM*) via the **request** table. The UNIX `crontab` file periodically starts *polling,* which writes sample requests to the **request** table. The **request** table also serves as the interface to the DACS, which controls the data flow for outbound request messages. Another interface to the DACS is *tuxshell,* which calls the message formatting process *dispatch* of the Retrieve Subsystem. In turn, *dispatch* reports its exit status to *tuxshell*, which updates the *state* of processed requests in the **request** table. The Retrieve Subsystem interfaces with the Message Subsystem and passes data request messages by writing message files to a permanent storage directory and by inserting information into the **msgdisc** and **msgdest** tables. It also sends a message to *AutoDRM* if data for redirected user requests have been received. Finally, the **wfdisc** table is another interface to the

Automatic and Interactive Processing CSCIs. The **wfdisc** table is used by the Retrieve Subsystem to verify successful data retrieval and by the Automatic and Interactive Processing CSCIs to access received data after successful data retrieval.

Interface with External Users

The Message Subsystem [IDC7.4.2] is the interface to IMS stations, which receive request messages from the IDC and respond with data messages.

Interface with Operators

The Retrieve Subsystem is designed to run automatically and without regular operator intervention. *WaveAlert* is the only process of the Retrieve Subsystem that runs continuously. All other processes are either called as child processes of other processes or started from the UNIX utility *cron*. *WaveAlert* starts up and shuts down together with the Message Subsystem. All processes, with the exception of *polling,* write log files to the general logging area. The operator can inspect the status of the Retrieve Subsystem in the log files and track the processing *state* of data requests in the database tables. A more convenient tool for the operator to inspect the status of the **request** table is the graphical *RequestFlow* display, which is derived from *WorkFlow*.

Chapter 3: Detailed Design

This chapter describes the detailed design of the Retrieve Subsystem and includes the following topics:

- Data Flow Model
- Processing Units
- Database Description

Chapter 3: Detailed Design

DATA FLOW MODEL

Figure 5 shows the data flow model of the Retrieve Subsystem. Data requestors insert their requests for auxiliary data into the **request** table. The Retrieve Subsystem itself acts as a requestor for sample requests to test station availability. These sample requests are inserted by *polling*. The DACS checks the **request** table in periodic intervals for new requests. If new requests are found, the DACS component *tuxshell* calls *dispatch* to format GSE2.0 or IMS1.0 request messages with the information from the **request** table. *dispatch* calls *MessageSend* as a child process, which adds the email header, STOP lines, plus a digital signature (if configured to do so), and writes the request messages to a permanent storage directory. *MessageSend* also inserts information about the stored messages in the **msgdisc** and **msgdest** database tables. At this point the Message Subsystem [IDC7.4.2] takes over the role to send the request messages to the IMS stations, which respond automatically via their *AutoDRM*s, and return data messages if the requested data are available and the stations and data links are operational. The Message Subsystem parses incoming data messages and stores waveform files. It also inserts descriptive information about received data in the **wfdisc** table. *WaveAlert* compares the **request** table to the **wfdisc** and **outage** tables in periodic intervals to verify if requested data have been successfully received. If no data have been received after a configurable time (*cleanup-time*) after a request has been sent, the Retrieve Subsystem sends the request message again. If data are still not retrieved, the Retrieve Subsystem gives up the request after a second time interval (*max-submit-time*).

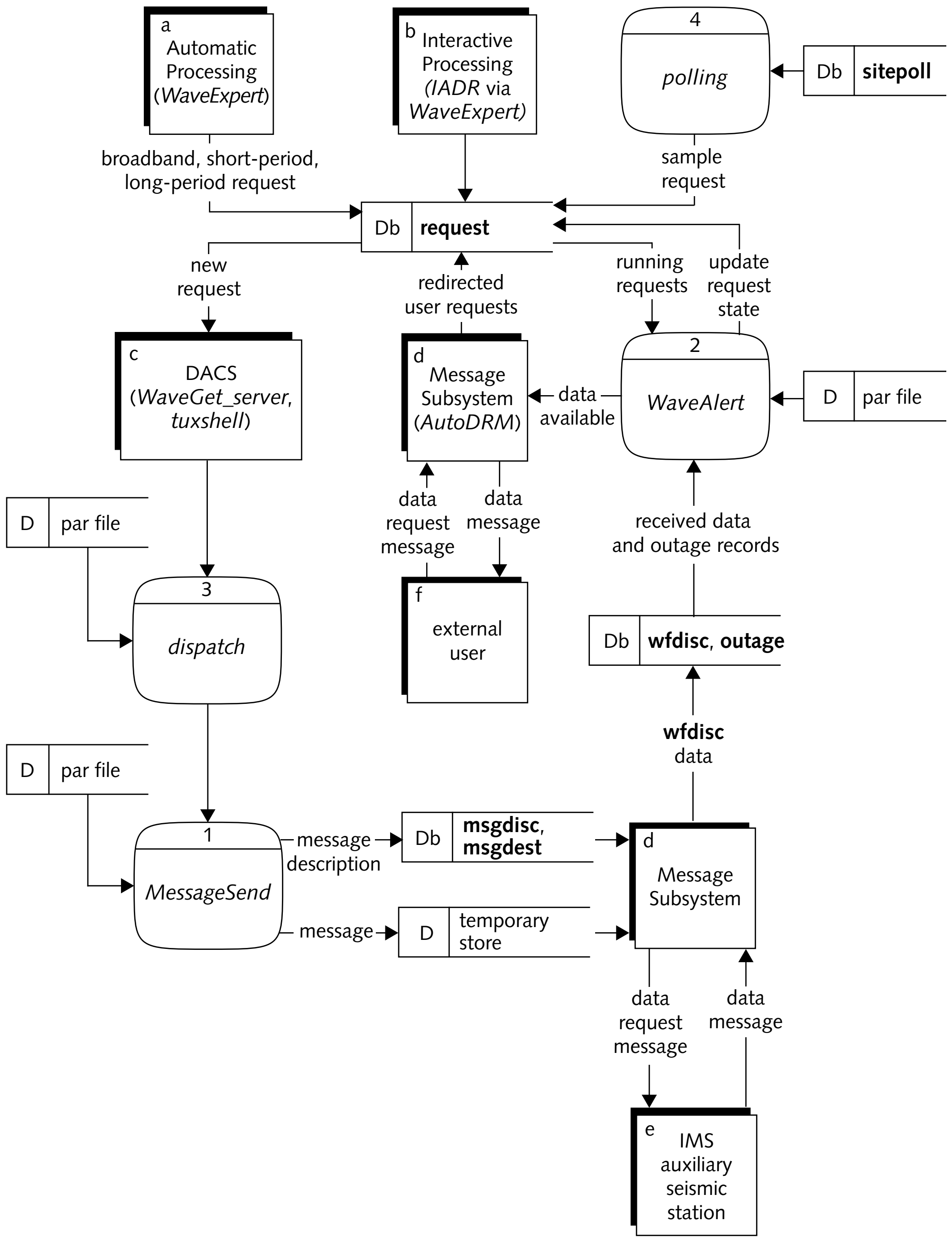


FIGURE 5. DATA FLOW MODEL OF RETRIEVE SUBSYSTEM

PROCESSING UNITS

The Retrieve Subsystem consists of the following processing units:

- *polling*
- *dispatch*
- *MessageSend*
- *WaveAlert*

The following paragraphs describe the design of these units, including any constraints or unusual features in the design. The logic of the software and any applicable procedural commands are also provided.

polling

polling is a lightweight Perl script that requests short data samples from auxiliary stations to test station availability. Entries are made in the **request** table with the *state* of `requested`. *polling* is started from *cron*.

Input/Processing/Output

polling reads entries from the **sitepoll** table to determine which stations and channels to request. It writes entries for a one minute interval of data for all these stations and channels to the **request** table. No log file is written and no parameters are used, except for a database connect string.

Control

polling is executed from *cron* at an interval appropriate for monitoring auxiliary station availability. It terminates after the sample requests are written to the **request** table.

Interfaces

polling is called from *cron*. It reads from the **sitepoll** table and inserts short data requests into the **request** table, then *polling* terminates. Only database information is exchanged. The successful writing of requests can be checked in the **request** table and on the graphical *RequestFlow* display.

Error States

polling does not use particular error codes. It uses the *ora_select* package to access the database, which returns standard ORACLE error codes. If *polling* cannot be executed, *cron* sends a notification to the UNIX user. The successful execution of polling can be monitored in the **request** table and on the graphical *RequestFlow* display, which is derived from *WorkFlow*.

dispatch

dispatch maps a waveform request into the body of a properly formed and addressed GSE2.0 or IMS1.0 format request message. *dispatch* executes *MessageSend* to add the header and `STOP` lines plus a digital signature to the request message, and handles the insertion of the waveform request into the Message Subsystem.

Input/Processing/Output

dispatch gets the start times, end times, station, and channel of the waveform request from the command line arguments. All parameters are described in the *dispatch* man page(s). The formatted request message is passed directly in the command line to *MessageSend,* which is called as a child process.

dispatch writes separate log files for each station from which data are requested. These log files also contain output from the parent process, *tuxshell*.

Control

dispatch is started by *tuxshell,* which is a component of the DACS. *dispatch* terminates after the request is mapped to a formatted request message and *MessageSend* is executed.

Interfaces

dispatch reads arguments from the command line and parameters from the par file. It then maps the data request to a formatted request message and passes this message to *MessageSend,* which is called as a child process. Logging information is written to the same log files, which are used for corresponding *tuxshell* output.

Error States

dispatch exits with the Perl `die` command if required parameters are not defined, and it writes a message to its log file. If the child process returns a non-zero status, *dispatch* writes that status code to its log file and exits with *status* = `1`.

MessageSend

MessageSend supports the gathering of time-shared data from stations of the auxiliary seismic network, accepts data specifications from *dispatch*, formats the specifications into messages that adhere to the standard protocols [IDC3.4.1Rev2], and writes the messages requesting data to the filesystem from where they are picked up by the Message Subsystem.

MessageSend assumes that the messages are received by an *AutoDRM* process at the IMS station after they have been sent by the Message Subsystem.

Input/Processing/Output

MessageSend reads a message from standard input and writes the entire message to disk under the *msgdir* directory (*msgdir* is a parameter in `MessageSend.par`). Entries are made in the **msgdisc** and **msgdest** tables, with the status of `NEW` and `PENDING` for the new message. *dispatch* writes the request portion of the message, and then *MessageSend* creates the header and STOP lines of the message as specified in the formats and protocols for messages [IDC3.4.1Rev2]. *MessageSend* also adds a digital signature if configured to do so. All parameters are described in the *MessageSend* man page(s).

Control

MessageSend is executed by *dispatch* and terminates after the message read from standard input is processed.

Interfaces

MessageSend is invoked as a child process of *dispatch*. It reads the body of the message from standard input and adds the header and STOP lines plus a digital signature to the message as specified in [IDC3.4.1Rev2]. The message is written to a permanent directory (*msgdir*), and records are written to the **msgdisc** and **msgdest** tables. All log information is written to the filesystem, as specified in the man page(s) for *liblogout*.

Error States

If *MessageSend* cannot access the database or cannot write the message to disk, it exits with a non-zero status and the parent process handles the error.

WaveAlert

WaveAlert compares data records from the **wfdisc** table and entries in the **outage** table to request entries in the **request** table; it then updates the *state* of requests based on this comparison and sends a message to *AutoDRM* if data for indirect *AutoDRM* requests have been received.

Input/Processing/Output

WaveAlert reads and compares entries from the **request**, **wfdisc**, and **outage** tables. It updates the *state* attribute in the **request** table. The *state* values used in the **request** table are described in Table 2. If all requested data have been received, the requests are set to the final *state*, `done-successful`, and if the requestor is *AutoDRM* a message is sent to *AutoDRM* so it can respond to the external user who originally requested the data. Requests for which no data have been received after a configurable time interval (*max-submit-time*) are set to `done-no-data`; and requests for which data have been partially received are set to `done-partial`. All three *state* values are final; the Retrieve Subsystem stops processing requests that are in one of these states. However, *WaveAlert* also uses a *state* `retry` to update requests for which no data have been received. Requests in *state* `retry` are picked up again by the Retrieve Subsystem, and the request message is resent to the IMS station. The time interval to retry such unanswered requests is configurable (*cleanup-time*) and in general should be shorter than the time interval to finally update requests to *state* `done-no-data` (*max-submit-time*). All parameters are described in the *WaveAlert* man page(s).

TABLE 2: STATES OF REQUESTS IN REQUEST TABLE

State	Description	Author
`requested`	Initial *state* for new requests, waiting to be picked up by *WaveGet*.	all requestors (for example: *WaveExpert*, *polling*, and so on)
`queued`	*WaveGet_server* has picked up the request and sent an IPC message to the DACS, request is in the dispatch-queue.	*WaveGet_server*
`running`	*dispatch* and *MessageSend* have processed the request and passed it on to the Message Subsystem, which sends it to the IMS station. The Retrieve Subsystem is waiting for a response from the station.	*tuxshell-dispatch*

TABLE 2: STATES OF REQUESTS IN REQUEST TABLE (CONTINUED)

State	Description	Author
`dispatch-failed`	*dispatch* failed to format the request, or *WaveGet_server* updated the request to *state* `dispatch-failed` after having retried the request a *max-retry* number of times unsuccessfully.	*WaveGet_server*, *tuxshell-dispatch*
`done-success`	Data have been successfully retrieved.	*WaveAlert*
`done-partial`	Data have been partially retrieved after *cleanup-time* and the request will not be retried.	*WaveAlert*
`retry`	Data have not been retrieved after *cleanup-time* and request is not older than *max-submit-time*, so it is set to *state* `retry` by *WaveAlert* and will be picked up again by *WaveGet_server*.	*WaveAlert*
`dispatch-retry`	If *dispatch* fails to format the request, *tuxshell* will update its *state* to `dispatch-retry` and call *dispatch* again before it finally sets the *state* to `dispatch-failed`.	*tuxshell-dispatch*
`done-no-data`	Data have not been retrieved after *max-submit-time*.	*WaveAlert*

Control

WaveAlert is the only continuously running process of the Retrieve Subsystem. *WaveAlert* is started and shut down together with the Message Subsystem by the operator or, alternatively, by a shell script, which starts the Message and Retrieve Subsystem at boot time.

After comparing and updating the database tables, *WaveAlert* sleeps for *sleep-time* seconds before it performs the next check in the database.

Interfaces

WaveAlert reads from the **request**, **wfdisc**, and **outage** tables. It compares data and request records and decides whether requests have been successful or not. *WaveAlert* uses the result of this decision to update the *state* attribute in the **request** table. Messages for *AutoDRM* (in case data for redirected user requests have been received) are written to the common message staging directory (*dirname* in `msgs.par`). All log information is written to the filesystem, as specified in the man page(s) for *liblogout*.

Error States

WaveAlert exits with a non-zero status if required parameters are missing at startup or if the database cannot be opened. All diagnostic information is written to the log files.

DATABASE DESCRIPTION

This section describes the database design and schema for those tables used by the Retrieve Subsystem. The Retrieve Subsystem uses the database primarily to track waveform requests, store records of request messages that are passed to the Message Subsystem, check for retrieved data, and update the *state* of requests. *polling* also uses the database to look for stations and channels for which sample requests should be generated. *MessageSend* and *WaveAlert* access the database using the Generic Database Interface (GDI); *polling* accesses the database using the *ora_select* package; *dispatch* does not access the database.

Database Design

The entity-relationship model of the schema is shown in Figure 6. The fundamental database table for the Retrieve Subsystem is the **request** table. The **request** table is used by data requestors to create waveform requests. The Retrieve Subsystem uses the **request** table to track the status of requests, compare received data with requests, and resend requests for which no data have been received. One special requestor is *polling,* which uses the **sitepoll** table to extract information about auxiliary seismic stations and channels for which to create sample requests. *Message-Send* writes information about formatted request messages to the **msgdisc** and **msgdest** tables, which serve as interfaces to the Message Subsystem [IDC7.4.2]. The Message Subsystem reads those tables and sends request messages to the IMS stations. The **wfdisc** and **outage** tables are used to store information about data received from IMS stations and about station outages. *WaveAlert* reads from these tables and compares them to active entries in the **request** table to check whether requests have been successfully satisfied or should be retried. Finally, the **lastid** table is used by all IDC subsystems to record the latest used values of numeric keys used in other database tables. The Retrieve Subsystem uses the numeric keys *reqid* in the **request** table and *msgid* and *msgdid* in the **msgdisc** and **msgdest** tables. *reqid* is only incremented by data requestors when they create new request entries; *msgid* and *msgdid* are incremented by the Retrieve Subsystem when it writes new message entries to be sent out by the Message Subsystem.

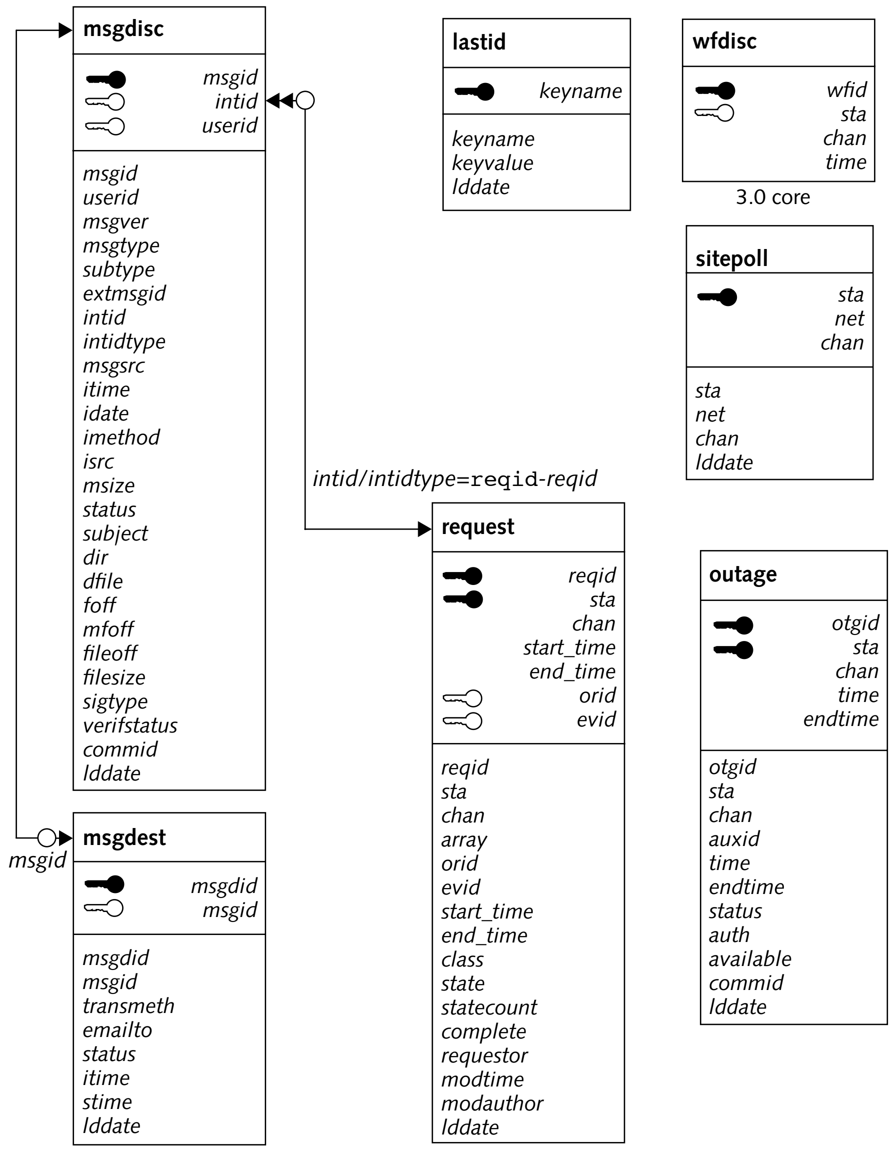


FIGURE 6. RETRIEVE SUBSYSTEM TABLE RELATIONSHIPS

Database Schema

Table 3 shows the usage of database tables by the Retrieve Subsystem. For each table used, the third column shows the purpose for reading or writing each attribute.

TABLE 3: DATABASE USAGE BY RETRIEVE SUBSYSTEM

Table	Action	Usage of Attributes
request	reads	• all attributes to format request messages to be sent to IMS stations • all attributes to compare active requests to data received from IMS stations
	writes	• *state* to update the state of requests • all attributes to define sample station-channel-time intervals of data to be retrieved from auxiliary stations
sitepoll	reads	• all attributes to obtain information about stations and channels for which to create sample requests
msgdest	writes	• all attributes to store information about the destination address, delivery method, and time when a request message was sent out
msgdisc	writes	• all attributes to create new request messages to be sent by the Message Subsystem to IMS stations
outage	reads	• *sta*, *chan*, *time*, and *endtime* to obtain information about requests which cannot be satisfied
lastid	reads	• *msgid* and *msgdid* to obtain next available numeric value for these keys
	writes	• *msgid* and *msgdid* to store new last value for these sequential keys
wfdisc	reads	• *sta*, *chan*, *time*, and *endtime* to compare received data intervals to active waveform requests

References

The following sources supplement or are referenced in this document:

[DOD94a]	Department of Defense, "Software Design Description," *Military Standard Software Development and Documentation,* MIL-STD-498, 1994.
[Gan79]	Gane, C., and Sarson, T., *Structured Systems Analysis: Tools and Techniques*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1979.
[IDC3.4.2]	Science Applications International Corporation, *Formats and Protocols for Continuous Data*, SAIC-98/3005, 1998.
[IDC3.4.1Rev2]	Science Applications International Corporation, Veridian Pacific-Sierra Research, *Formats and Protocols for Messages, Revision 2*, SAIC-00/3005, PSR-00/TN2829, 2000.
[IDC3.4.3]	Science Applications International Corporation, *Formats and Protocols for Continuous Data CD-1.1*, SAIC-00/3026, 2000.
[IDC5.1.1Rev2]	Science Applications International Corporation, Veridian Pacific-Sierra Research, *Database Schema, Revision 2*, SAIC-00/3057, PSR-00/TN2830, 2000.
[IDC6.5.2Rev0.1]	Science Applications International Corporation, *Distributed Application Control System (DACS) Software User Manual, Revision 0.1*, SAIC-00/3038, 2000.
[IDC7.4.2]	Science Applications International Corporation, Pacific-Sierra Research, Inc., *Message Subsystem*, SAIC-98/3003, 1998.

Glossary

A

analyst

Personnel responsible for reviewing and revising the results of automatic processing.

architecture

Organizational structure of a system or component.

architectural design

Collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.

archive

Single file formed from multiple independent files for storage and backup purposes. Often compressed and encrypted.

attribute

(1) Database column. (2) Characteristic of an item; specifically, a quantitative measure of a S/H/I detection such as azimuth, slowness, period, or amplitude.

C

channel

Component of motion or distinct stream of data.

child process

UNIX process created by the *fork* routine. The child process is a snapshot of the parent at the time it called *fork*.

CMR

Center for Monitoring Research.

command

Expression that can be input to a computer system to initiate an action or affect the execution of a computer program.

commercial-off-the-shelf

Terminology that designates products such as hardware or software that can be acquired from existing inventory and used without modification.

component

(1) One dimension of a three-dimensional signal; (2) The vertically or horizontally oriented (north or east) sensor of a station used to measure the dimension; (3) One of the parts of a system; also referred to as a module or unit.

Comprehensive Nuclear-Test-Ban Treaty Organization

Treaty User group that consists of the Conference of States Parties (CSP), the Executive Council, and the Technical Secretariat.

Computer Software Component

Functionally or logically distinct part of a computer software configuration item; possibly an aggregate of two or more software units.

Computer Software Configuration Item

Aggregation of software that is designated for configuration management and treated as a single entity in the configuration management process.

configuration

(1) (hardware) Arrangement of a computer system or components as defined by the number, nature, and interconnection of its parts. (2) (software) Set of adjustable parameters, usually stored in files, which control the behavior of applications at run time.

COTS

See commercial-off-the-shelf.

CSC

See Computer Software Component.

CSCI

See Computer Software Configuration Item.

CTBTO

See Comprehensive Nuclear-Test-Ban Treaty Organization.

D

DACS

See Distributed Application Control System.

data flow

Sequence in which data are transferred, used, and transformed during the execution of a computer program.

Distributed Application Control System

This software supports inter-application message passing and process management.

DSA

Digital Signature Algorithm.

E

email

Electronic mail.

entity-relationship (E-R) diagram

Diagram that depicts a set of entities and the logical relationships among them.

event

Unique source of seismic, hydroacoustic, or infrasonic wave energy that is limited in both time and space.

execute

Carry out an instruction, process, or computer program.

G

GB

See gigabyte.

GDI

Generic Database Interface.

gigabyte

Measure of computer memory or disk space that is equal to 1,024 megabytes.

GSETT-3

Group of Scientific Experts Third Technical Test.

I

IDC

International Data Centre.

IMS

International Monitoring System.

IPC

Interprocess communication. The messaging system by which applications communicate with each other through *libipc* common library functions. See *tuxshell*.

L

libgdi

Library containing functions for RDBMS access.

M

magnitude

Empirical measure of the size of an event (usually made on a log scale).

MB

See megabyte.

megabyte

1,024 kilobytes.

O

ORACLE

Vendor of the database management system used at the PIDC and IDC.

P

par

See parameter.

parameter

User-specified token that controls some aspect of an application (for example, database name, threshold value). Most parameters are specified using [*token* = *value*] strings, for example, `dbname=mydata/base@oracle`.

parameter (par) file

ASCII file containing values for parameters of a program. Par files are used to replace command line arguments. The files are formatted as a list of `[`*token* `=` *value*`]` strings.

parent process

UNIX process that creates a child process by the *fork* routine. The child process is a snapshot of the parent at the time it called *fork*.

parse

Decompose information contained in a set of data.

PIDC

Prototype International Data Centre.

post-location processing

Software that computes various magnitude estimates and selects data to be retrieved from auxiliary stations.

process

Function or set of functions in an application that perform a task.

S

schema

Database structure description.

script

Small executable program, written with UNIX and other related commands, that does not need to be compiled.

seismic

Pertaining to elastic waves traveling through the earth.

SEL1

Standard Event List 1; S/H/I bulletin created by total automatic analysis of continuous timeseries data. Typically, the list runs about two hours behind real time.

SEL2

Standard Event List 2; S/H/I bulletin created by totally automatic analysis of both continuous data and segments of data specifically down-loaded from stations of the auxiliary seismic network. Typically, the list runs about six hours behind real time.

Solaris

Name of the operating system used on Sun Microsystems hardware.

States Parties

Treaty user group who will operate their own or cooperative facilities, which may be NDCs.

station

Collection of one or more monitoring instruments. Stations can have either one sensor location (for example, BGCA) or a spatially distributed array of sensors (for example, ASAR).

subsystem

Secondary or subordinate system within the larger system.

T

time series

Time ordered sequence of data samples. Typically a waveform or derived from waveforms, such as a beam.

Tuxedo

Transactions for UNIX Extended for Distributed Operations.

tuxshell

Process in the Distributed Processing CSCI used to execute and manage applications. See IPC.

U

UNIX

Trade name of the operating system used by the Sun workstations.

W

waveform

Time-domain signal data from a sensor (the voltage output) where the voltage has been converted to a digital count (which is monotonic with the amplitude of the stimulus to which the sensor responds).

Web

World Wide Web; a graphics-intensive environment running on top of the Internet.

wfdisc

Waveform description record or table.

workstation

High-end, powerful desktop computer preferred for graphics and usually networked.

Index